REMARKS

In response to the Office Action mailed December 21, 2006, Applicant respectfully requests reconsideration and continued examination. By this amendment, Applicant cancels claims 26-30 without prejudice or disclaimer. Claims 1-24 are pending in the application of which claims 1, 11, 21, and 22 are independent. Claims 1-9, 11-19, and 21-24 are amended in the foregoing amendments to clarify Applicant's invention. No new matter is introduced by these amendments.

Double Patenting Rejection

In the Office Action mailed December 21, 2006, the Examiner maintained the double patenting rejection of the Office Action mailed June 15, 2005, where claims 1-25 were provisionally rejected under the judicially created doctrine of obviousness-type double patenting over claims 1-10, 13-22, and 25-28 of copending Application No. 09/759,697. Applicant requests that the Examiner continue to hold the double patenting rejection in abeyance until claims 1-24 are otherwise allowable.

Rejections Under 35 U.S.C. § 103

In the Office Action mailed December 21, 2006, the Examiner rejected claims 1, 11, 21, and 22 over Linked list code examples from Data Structures and other Objects Using C++ by Main and Savitch (1997) listed in the cs.appstate.edu website (Index of... examples; "Bag Implementation Using Linked Lists," 1998), hereinafter "Main," in view of US Patent 6,045,585, hereinafter "Blainey." The Examiner rejected claims 1-8, 11-18, 21, 22-24 and 26-30 as unpatentable over McLennan ("Object-Oriented Programming with [incr Tcl] Building Mega-Widgets with [incr Tk]," 1996) in view of

Main, and further in view of Blainey. The Examiner also rejected claims 9, 10, 19, and 20 as being unpatentable over McLennan in view of Main, in further view of Blainey, and in further view of Hostetter et. al. ("Curl: A Gentle Slope Language for the Web," World Wide Web Journal, Spring 1997, art of record) hereinafter "Hostetter." Applicant respectfully traverses the rejections under 35 U.S.C. § 103 because the Examiner has not established a prima facie case of obviousness.

The Office Action fails to establish a prima facie case of obviousness for at least the reason that the cited references, taken alone or combined, fail to teach each and every element of independent claim 1. *See* MPEP § 2143.03 ("To establish prima facie obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974).").
Specifically, the Examiner uses Blainey to make up for the noted deficiencies of Main and McLennan, which do not disclose or suggest, "during compilation, determining whether at least one of the expressions accesses one of (a) the selected field value or (b) the selected option value; when it is determined that the selected field value is being accessed, compiling the expression into a first code for accessing the selected field value; and when it is determined that the selected option value is being accessed, compiling the expression into a second code for accessing the selected option value." See Office Action at 3 and 5. However, Applicant maintains that Blainey fails to teach or suggest this element of Applicant's claim 1.

The Examiner contends that Blainey teaches a "data type check" performed by a compiler, and that through this teaching at col. 2 lines 27-40, Blainey discloses this element of Applicant's claim 1. See Office Action at 3 and 5. However, the data type

check in Blainey is not comparable to this element of Applicant's claim 1 for at least two reasons. First, Blainey discloses that type checking by the compiler determines the data type (integer vs. float, for example) of a declared variable. However, in Applicant's claim 1, the determination is of the property type or class member type (option vs. field, for example). The property type and data type are not analogous since the properties themselves have data types. See Applicant's Specification at 2, "type description of the option value" and Applicant's Specification at 5, "declared type of field x." Blainey does not disclose or suggest the same determination as in Applicant's claim 1 since Blainey does not disclose or suggest determining, during compilation, the property type. The type checking in Blainey would not result in a determination of whether a property is an option or field, and therefore combining Blainey with Main or McLennan would not result in Applicant's claimed invention.

Second, Blainey does not disclose or suggest compiling the expression into a first code or a second code depending on the determination of whether the accessed value is a field value or an option value. Blainey does not even disclose or suggest compiling an expression accessing the different data types differently much less disclose or suggest compiling expressions accessing different property types differently.

Moreover, none of the references cited by the Examiner disclose or suggest, "accessing a selected field value in the instance using a first single program code expression, the first single program code expression comprising an operator and the selected field name, and accessing a selected option value in the instance using a second single program code expression, the second single program code expression comprising the operator and the selected option name," as recited in Applicant's claim

1. In Main, the many_nodes field value is accessed using the dot operator (i.e. addend.may_nodes). See Main at 2, "many_nodes = source.many_nodes." However, nowhere does Main disclose or suggest that one of the data items (which the Examiner says are analogous to options) could be referenced in the same manner, using the name of the data item and the dot operator. See Office Action at 3. Instead, Main shows examples that access the data items "target" or "entry" through the use of a function call to traverse the linked list of data items. See Main at 3, "cursor = list_search(head_ptr, target)." Nowhere in Main is a data item referenced by name using the dot operator. McLennan, Blainey, and Hostetter also do not disclose or suggest "accessing a selected field value in the instance using a first single program code expression, the first single program code expression comprising an operator and the selected field name, and accessing a selected option value in the instance using a second single program code expression, the second single program code expression comprising the operator and the selected option name," as recited in Applicant's claim 1.

For the reasons stated above, the rejection under 35 U.S.C. § 103 of claim 1 as well as the rejection of claims 11, 21, and 22 which contain similar elements and were rejected on the same rationale, should be withdrawn. In addition, dependent claims 2-10, 12-20, and 23-24, are allowable under 35 U.S.C. § 103 at least since they depend from claims 1, 11, and 22.

In view of the foregoing amendments to claims and remarks, Applicant respectfully requests the reconsideration of this application and the timely allowance of the pending claims.

Please grant any extensions of time required to enter this response and charge any additional required fees to our deposit account 06-0916.

Respectfully submitted,

FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L.L.P.

Dated: 14 June 2007

By: Maura K. Moran

Maura K. Moran
Reg. No. 31,859